View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by BOP Serials

Natural language processing tools for computer assisted language learning

Anne Vandeventer Faltin (Geneva)

Abstract

This paper illustrates the usefulness of natural language processing (NLP) tools for computer assisted language learning (CALL) through the presentation of three NLP tools integrated within a CALL software for French. These tools are (i) a sentence structure viewer; (ii) an error diagnosis system; and (iii) a conjugation tool.

The sentence structure viewer helps language learners grasp the structure of a sentence, by providing lexical and grammatical information. This information is derived from a deep syntactic analysis. Two different outputs are presented.

The error diagnosis system is composed of a spell checker, a grammar checker, and a coherence checker. The spell checker makes use of alpha-codes, phonological reinterpretation, and some *ad hoc* rules to provide correction proposals. The grammar checker employs constraint relaxation and phonological reinterpretation as diagnosis techniques. The coherence checker compares the underlying "semantic" structures of a stored answer and of the learners' input to detect semantic discrepancies.

The conjugation tool is a resource with enhanced capabilities when put on an electronic format, enabling searches from inflected and ambiguous verb forms.

1 Introduction

Computer assisted language learning (CALL) emerged in the early days of computers. Since the early 1960's, CALL software was designed and implemented. In the drill-and-practice days of language teaching methodology, what could, better than a computer, repeatedly and relentlessly propose exercises to the learners? Since then, advances in second language acquisition (SLA) have helped the shift of interest from repetitive exercises to more communicative tasks.

The range of exercises offered in CALL software has always been constrained by the kind of feedback judged necessary for the learning process and its availability on a computer. Thus, too often, exercises tend to elicit answers easily classified as right or wrong. This leads to all sorts of multiple-choice-questions, fill-in-the-blanks, and similar exercise types which computers can very easily correct automatically. For slightly more complex exercises requiring the learners to write a few words, either no feedback is provided or pattern matching

ISSN 1615-3014

techniques are used, requiring exercise authors to enter many different correct and wrong answers. This very tedious and time consuming task yet only provides appropriate feedback when the learner types in one of the expected answers.

True communicative tasks are in need of more "intelligent" devices to provide appropriate feedback to the users. Natural language processing (NLP) tools seem to be the obvious answer. NLP tools can be advantageously used for the correction of free production exercises where they can diagnose, in a generic way, many kinds of mistakes. Besides exercise correction, NLP tools can also be used as additional help resources within CALL software, to enable the learners to get more of the content materials: written texts can be listened to with the help of a speech synthesizer, sentences can be analyzed with a sentence structure viewer, verb conjugations can be verified with a conjugation tool, to give only a few examples.

A very complete and interesting introduction to NLP in the CALL context can be found in (Nerbonne 2003), which moreover contains very extensive bibliography. Collections of papers discussing the use of NLP for CALL include (Schulze *et al*. 1998), (Holland *et al*. 1995), and (Swartz/Yazdani 1992), among others.

In this paper, we illustrate the usefulness of NLP for CALL based on the presentation of three NLP tools[1] designed for French and integrated within a CALL software, namely: (i) a sentence structure viewer displaying information on the words of a sentence and on its principal constituents (section 2); (ii) an error diagnosis system, including a spell checker, a grammar checker, and a coherence checker[2] (section 3); and (iii) a conjugation tool (section 4). Section 5 concludes this paper.

2 Sentence structure viewer

A sentence structure viewer is a resource tool provided to help language learners grasp the structure of a sentence. This can help them understand the sentence better, correct some of their mistakes, or appropriate grammatical subtleties of the language.

Our sentence structure viewer has two different outputs, a linear color display (section 2.2) as well as a hierarchical structure tree display (section 2.3). However, the core of the tool is exactly the same for both outputs: a syntactic parser, described in section 2.1, is used to provide a rich syntactic analysis of the sentences. The information resulting from the parsing process is screened and displayed differently by each distinct output.

---

[1] These three tools were designed, implemented, and/or improved at the Department of Linguistics of the University of Geneva, essentially as part of the FreeText project. The FreeText project received financial support from the European Commission in the IST programme of the 5th framework-programme, contract IST-1999-13093. The Swiss participation was funded by the Swiss Federal Office for Education and Science, contract 99.0049.

[2] In the FreeText project, this coherence checker was called *'semantic' checker*.

2.1 Syntactic parsing

Syntactic analysis is provided by a pre-existing syntactic analyzer, the Fips parser. Fips has been fully described elsewhere (Wehrli 97, Laenzlinger/Wehrli 1991), but we nevertheless very briefly present the main characteristics of this parser as they are important both for the sentence structure viewer and for the error diagnosis system (section 3).

Fips is a syntactic parser based on the Government and Binding theory (Chomsky 1981, among others). The main goal of the parser is to provide syntactic structures for input sentences in French. Grammaticalness of the input sentences is assumed, as the parser was not developed at first for the CALL context.

The parsing algorithm follows the right corner strategy (Wehrli 1997: 220). The four main principles of this strategy are the following

1. Ascending strategy, that is data driven;
2. Iteratively, from left to right, one reads a new element and one tries to combine it to a maximally developed structure from the left context;
3. The left structure specifies a list of active nodes on which new elements can attach;
4. All possible alternatives are considered in parallel.

A lexical analysis is first performed to segment the input. For each lexical element, a projection of the same category is created. They are stored in a graph of well-formed (although possibly partial) constituents. The parser tries to combine two adjacent projections, *a* and *b*. *a* can attach as the specifier of *b* (attachment to the left), or *b* as a complement of an active node of *a* (attachment to the right). An active node is a position on the right edge of a tree structure which is available to receive new complements. The list of active nodes is kept and updated at each new attachment in order to facilitate and speed up the process of attachment to the right. Each new combination is stored in the graph. Possible attachments are treated in parallel, with the help of heuristics to restrain the size of the set of constituents to be considered for attachment.

At the end of the parsing process, a very rich information is available. The standard output of the parser, a tree-like syntactic structure represented linearly by labeled bracketing, provides only some of the elements of this rich information. The sentence structure viewer, for its two distinct output modes, makes use of a much wider range of the available information.

Scores are given to each partial and complete structure in order to classify alternatives during parsing and to choose which full analysis to display when parsing is successful. When no complete analysis is found by the parser, partial analyses of the input are presented which, put side by side, cover the whole sentence. Thus, an output is provided even when the input does not result in a successful parse. The presence of partial analyses implies that the input is either ungrammatical or lies outside the capabilities of the parser.

Fips was designed to treat syntactically correct sentences, but its ability to produce an output even for incorrect sentences makes it suitable to treat learner input.

Figure 1: The linear color display.[3]

2.2 Linear color display

The linear color display, as illustrated in Figure 1, provides information on the main grammatical functions of the sentence. Subjects, predicates, attributes, direct and indirect objects, circumstantial complements, and agent complements focused on appear underlined in different colors. The choice of colors results from an adaptation of what is currently done within the Geneva primary school system.

Lexical categories can also be selected, in which case words are written in specific colored fonts on a neutral background. Thus, all the nouns, for example, can appear in blue. Besides indicating the lexical category through font coloring, more detailed lexical information is available on mouse browse-over. In this case, the canonical form of the word, its agreement features, and possibly mode and tense are displayed in a pop-up.

The linear color display is designed for use by language learners of all age groups and proficiency levels. Prior knowledge required to use this tool consists mainly on knowing how to distinguish the main grammatical functions and lexical categories. This, at least within the Geneva school system, is supposedly mastered by the end of primary school.

2.3 Hierarchical structure tree display

The hierarchical structure tree display requires more extensive knowledge of linguistics. Instead of indicating grammatical functions, it presents a syntactic tree of the sentence, following the grammatical formalism used by the syntactic parser, namely, an adapted version of Chomsky's Government and Binding theory (Chomsky 1981, among others).

[3] Figure 1 is a screenshot of the FreeText interface of the linear color display.

The tree has been simplified, however, to respond more closely to the needs of language learners: multiple branching is allowed, labels have been simplified and translated into French abbreviations,[4] in order to expose the learners to their target language. An example of this hierarchical structure tree representation is given in Figure 2 below. The red link indicates that the two elements are syntactically and semantically related. In the present case, the implicit subject of the embedded clause is the same as the overt subject of the main clause.

Such a representation requires more formal linguistic knowledge on the part of the users. It is therefore not easily accessible to everybody and people without any syntactic training might find themselves at a loss. However, this tree representation can prove more interesting for people with a linguistic background. Moreover, this output needs only minor modifications to be useful for teaching syntax itself, according to the Government and Binding framework, rather than a foreign language.

3 Error diagnosis system

An error diagnosis system must contain several components, the exact number of which depends on the sophistication of the complete system. Ideally, it should include a spell checker, a grammar checker, as well as a semantic/pragmatic checker. A coherence checker might also prove useful if the error diagnosis system is designed for exercise correction, rather than for inclusion in a word processing software. While spell and grammar checkers work intrasententially, a semantic/pragmatic checker should be able to diagnose errors in the wider context of the paragraph, text, and world. A coherence checker is limited to verifying that an exercise answer is in relation with the question asked. Most often, diagnosis systems are limited to the spell and grammar levels. The error diagnosis system described here is made of a spell checker identifying unknown words and proposing existing alternatives, a grammar checker handling some morpho-syntactic and syntactic errors, and a coherence checker dealing with the appropriateness of learners' answers for given questions. The coherence checker was designed but only partially implemented, due to lack of time.[5]

[4] The abbreviations used are as follows: Ph = *phrase* (sentence), GN = *groupe nominal* (noun phrase), GV = *groupe verbal* (verb phrase), Ph' = complementizer phrase, Conj = *conjonction* (conjunction), ei = $empty_i$, Inf = *infinitif* (infinitive), Gprép = *groupe prépositionnel* (preposition phrase), Prép = *préposition* (preposition), Dét = *déterminant* (determiner), N = *nom* (noun).

[5] The spell checker described in this section was designed and implemented at the Department of Linguistics of the University of Geneva. It is not the one which was used in the FreeText project, which was a commercial product. Moreover, as the coherence checker is not fully implemented, it was not integrated in the FreeText final error diagnosis system.

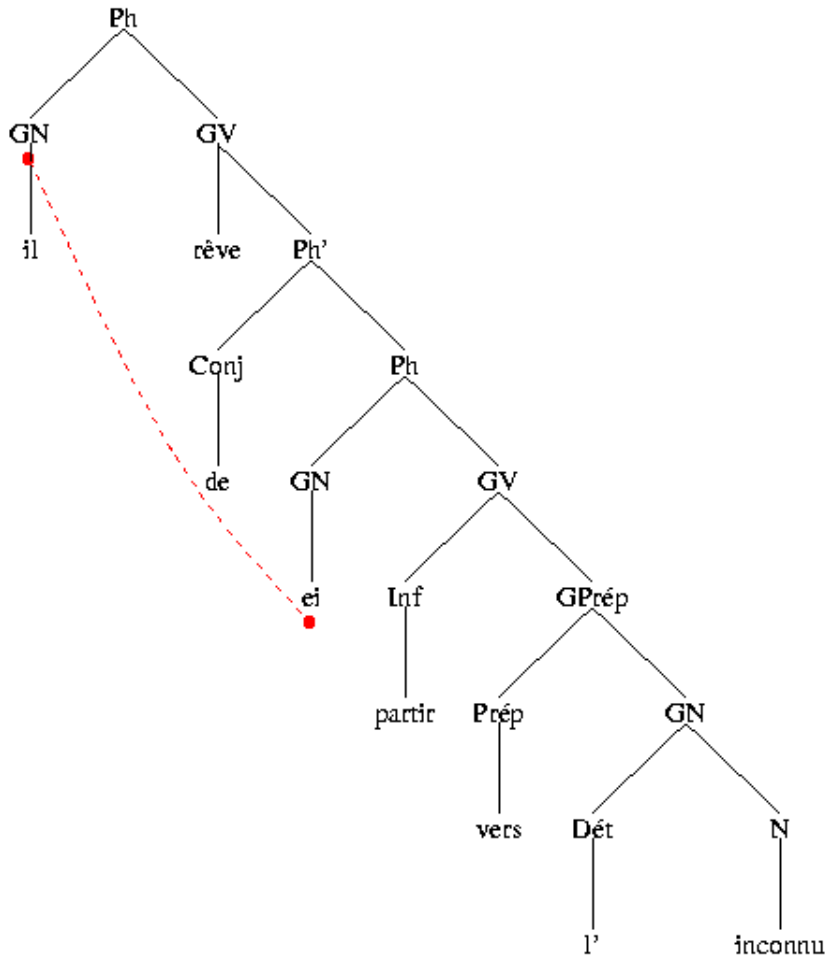

Figure 2: The hierarchical structure tree representation

Error diagnosis includes detecting errors, discovering their types and indicating their precise locations. Error correction involves diagnosing the errors and proposing correct alternatives for the erroneous parts. In CALL, in order to provide appropriate feedback, error diagnosis is more important than error correction. Indeed, knowing how to correct one's own errors based on appropriate information is part of the learning process. Providing a correction does not help the language learners as much to discover and to appropriate the correct forms and rules of the target language. Thus, the tool presented in this section focuses on diagnosis rather than correction, although correction proposals are available for spell checking and at times for grammar checking.

3.1 Spell checker

Traditional spell checkers, such as those found in most word processing software systems, contain a correction component and provide correction suggestions to their users. Detecting spelling errors is a rather straightforward task. One simply needs to verify that each word can be found in the system's lexicon. If it is not the case, then the existence of a spelling error is assumed and a search for correction proposals is initiated. Correction proposals are computed

using several distinct techniques. True diagnosis, however, indicating the precise reason of the error, is very rarely done.

The spell checker we designed follows this general consensus, although we are conscious of the need to be able, in the long run, to provide a diagnosis besides, or instead of, the correction proposals. Three different methods are used to retrieve correction proposals: alpha-codes, phonological reinterpretation, and *ad hoc* rules (pending the implementation of a morphological analyzer).

Much research has been done on spell checkers, bringing spell checkers up to a good standard some time ago. Thus, some of the literature on spell checking dates somewhat (Peterson 1980, Courtin *et al*. 1991, among others). However, little has been done, to my knowledge, on spell checkers targeting non-native learners.

3.1.1 Alpha-codes

An alpha-code is a simplified representation of a word, obtained by reordering the characters it is composed of, stripping the word of its diacritics, and getting rid of duplicated characters (Pollock/Zamora 1984). Words sharing a single alpha-code are similar in some ways. By computing the alpha-code of an unknown word and by retrieving all legal words which share the same alpha-code, one recuperates words which are close to the erroneous one. Spelling errors which can be treated by this method are inverted letters, incorrect diacritics (including multiple incorrect diacritics), as well as some missing or superfluous characters.

The erroneous word in example (1), containing two errors (a superfluous diacritic and a missing character), can find an appropriate correction through the alpha-code method.

(1a) *Je *jète l'emballage*.
I unknown-word the wrapping
(1b) jte
(1c) *jet*, *jeté*, *jetée*, *jette*
jet, thrown-away, thrown-away-feminine or jetty, throw-away
(1d) *Je jette l'emballage*.
I throw-away the wrapping.

The unknown word in (1a)[6] has the alpha-code shown in (1b) and can be associated with at least the correction proposals presented in (1c), which naturally all share the same alpha-code. The corrected sentence is given in (1d). Selecting the appropriate correction from the list in (1c) is often left to the users who are presented with a list of ordered correction proposals.
Some more examples of alpha-codes are given in Table 1. Each correction given has exactly the same alpha-code as its corresponding erroneous word. Moreover, note that both *rémunération* (remuneration) and *énumeration* (enumeration) have the same alpha-code, which implies that both correct words would be proposed as possible corrections if either one of them is misspelled in the manner presented in Table 1.

[6] The star * indicates erroneous words, phrases, or sentences.

The alpha-code technique, or one of its numerous variants, is very often used in commercial spell checkers and is not particularly oriented towards language learners.

Erroneous word	**Alpha-code**	**Correction**	**Translation**
**rémuneration*	mnrtaeiou	*rémunération*	remuneration
**ennumération*	mnrtaeiou	*énumération*	enumeration
**calandrié*	cdlraei	*calendrier*	calendar
**garcon*	cgnrao	*garçon*	boy
**pourait*	prtaiou	*pourrait*	could

Table 1: Examples of alpha-codes

3.1.2 Phonological reinterpretation

More innovative and more specifically geared towards language learners, phonological reinterpretation is another method to retrieve words similar to the incorrect one. This time, instead of basing the similarity criteria on the written characters, one uses the sounds of the language. An expert system (Gaudinat/Goldman 1998) is used to transform the written word into its phonological representation. It is then easy to search a lexicon indexed by pronunciation for all the words which sound alike. Another set of correction proposals is thus retrieved. Errors detected concern mostly the use of phoneme-to-grapheme rules which are not correct for the given word. Consider the illustration in (2).

(2a) **puit*
(2b) /pyi/
(2c) *puis*, *puits*
then, well

The erroneous word in (2a) has the phonological representation shown in (2b). Two words in French share this representation. They are given in (2c) and are the correction proposals which would be displayed to the users.

3.1.3 *Ad hoc* rules

Ad hoc rules allow our system to find adequate correction proposals for a morphological error which is not satisfactorily handled by the other two techniques. This is a very limited technique, used while waiting for the implementation of a morphological analyzer enabling our system to diagnose and correct the misuse of French inflection rules. Currently, *ad hoc* rules spot illegal words ending in *-als* or *-ails*.[7] These are supposedly made of a root word in *-al* or *-ail* and the regular plural marker *-s*, while these words should pluralize in *-aux*. Thus, the system replaces the incorrect ending by the supposedly correct one and checks that the newly composed word is indeed a true word before displaying it as a correction proposal. An example is given in (3) below. This limited technique is in place only so long as a true generic morphological treatment is not operational.

[7] Legal words in *-als* or *-ails*, such as *chacals* (jackals), do not trigger a correction attempt.

(3a) **généralsals*
(3b) *généraux*
general-PLURAL

3.1.4 Ordering the proposals

Results elicited from the three spelling correction techniques might coincide, but they are more likely to differ from each other. As too much information can be overwhelming, it is important to order the proposal list so that the most likely words appear on top. How to order the proposals should reflect the needs of the target user population. We present first the proposals retrieved by the *ad hoc* rules, followed by phonological reinterpretation, showing last the alpha-code recuperated words. This ordering brings to the top the corrections provided by the techniques which were designed for errors more likely to be committed by language learners.

3.2 Grammar checker

This grammar checker is based on the assumption that its input has previously undergone spell checking. Unknown words make it especially difficult for the grammar component of the diagnosis system to reach a likely analysis and must be filtered out by spell checking beforehand. Grammar checking is the second stage of the complete error diagnosis system.

An important feature of our grammar checker is that it provides a full syntactic analysis of the sentence as well as an error diagnosis. This full analysis is absolutely necessary for the following stage of the complete diagnosis process: coherence checking. Moreover, even without coherence checking, full syntactic analysis of an ungrammatical sentence is interesting. For example, it allows other NLP tools, such as the sentence structure viewer described in section 2, to function appropriately. Naturally, this limits to some extent the error diagnosis techniques which can be used: simply scanning the sentence for errors is not an option.

Research on automatic syntactic error detection has been plentiful. A few references are given here, among many others: Menzel/Schröder 1998, Schwind 1995, Vosse 1992, and Weinberg et al. 1995.

The grammar checker presented in this paper has its origins in the Fips parser described in section 2.1. Fips has been modified and adapted by the adjunction of two diagnosis techniques in order to detect specific error types in ungrammatical sentences while still providing full syntactic analysis whenever possible.

In this section, we present the error types treated in section 3.2.1 Description of the two syntactic diagnosis techniques, constraint relaxation (3.2.2) and phonological reinterpretation (3.2.3), are then given. Finally, we discuss some of the encountered problems (3.2.4).

3.2.1 Error types

Language learners make mistakes of many different error categories. While the obvious long term goal is to be able to diagnose all the kinds of errors produced, we have currently restricted our system to a fixed number of errors for detection and diagnosis at the grammatical level. The selection of which error types to treat was based on three criteria: (i) the frequency of the error categories in the FRIDA learner corpus (Granger 2003), (ii) an appreciation of their didactic importance, and (iii) the availability of appropriate diagnosis techniques. Thus, we settled on the following major error categories:[8] agreement in gender, number and person, verb and adjective complementation, adverb and adjective order, auxiliary selection, negation, euphony and homophony. A few examples are given in Table 2 below.

These error types are all detected and diagnosed via constraint relaxation, except for homophony errors, which are detected through phonological reinterpretation.

Error type	**Example**	**Correct version**
Gender agreement	*Les femmes sont *tolérés.* The women are tolerated-MASCULINE	*Les femmes sont tolérées.* Women are tolerated.
Verb complementation	*Je préfère *de partir.* I prefer of go	*Je préfère partir.* I prefer to go.
Adjective order	*Les sommes *vastes.* The sums vast	*Les vastes sommes.* The vast sums.
Auxiliary selection	*La femme *a resté un être fin.* The woman has stayed a being subtle	*La femme est restée un être fin.* *Women have remained subtle beings.*
Negation	**C'est pas un grand problème.* It is not a big problem	*Ce n'est pas un grand problem.* It is not a big problem.
Euphony	*Le *nouvel marché.* The new market	*Le nouveau marché.* The new market.
Homophony	**Ont pourrait conclure.* Have could conclude	*On pourrait conclure.* One could conclude.

Table 2: Samples of errors diagnosed by the grammar checker

3.2.2 Constraint relaxation

Constraint relaxation is a well-known error diagnosis technique of which there are several variants. To name only a few, Weischedel/Black (1980: 108) spoke of "failable" constraints, Matthews (1993) of principle violation, and Heinecke *et al.* (1998) of graded constraints. Although there are some differences between these versions of the technique, the underlying principles remain the same. Partial structures may combine to form a larger structure only if some constraints or conditions are met. For instance, a noun phrase can attach to a verb phrase

[8] Some other, less important, error types are also treated by the system but will not be discussed in this paper.

to form a complete clause only if they agree in number and person. It is the subject-verb constraint. When this constraint is relaxed, attachment is allowed even if the constraint is not satisfied, that is when the two phrases do not agree. Most systems using constraint relaxation can relax constraints separately from each other and relax them only when they cannot be satisfied. To be useful within a diagnosis system, the relaxed constraint must be marked on the structure, so that the type and position of the detected error can be indicated later on. Proper labeling of where the analysis has been able to proceed thanks to the specific relaxation of a constraint forms the basis of the error diagnosis (Felshin 1995).

Let us illustrate this technique with the sentence extract in example (4a), from the FRIDA corpus, containing a verb complementation error. This type of error is not usually treated by commercial grammar checkers because it is not a likely error for native speakers. The correct sentence is proposed in (4b).

(4a) **L'unification bénéficiera la Grande-Bretagne*.
The unification benefit-FUTURE the Great-Britain

(4b) *L'unification bénéficiera à la Grande-Bretragne*.
The unification benefit-FUTURE to the Great-Britain

Following the right corner parsing strategy (Wehrli 1997: 220), at one point during the parsing process, the incomplete determiner phrase (DP) headed by *la* (the) tries to attach as the complement of the verb *bénéficiera* (benefit-FUTURE) (5a). However, the current verb requires a prepositional phrase (PP) as a complement, rather than a DP. Thus, the verb complementation constraint is relaxed, allowing attachment of a DP as complement of *bénéficiera* with the adjunction of an error mark. The resulting temporary structure is given in (5b) and the remaining word (*Grande-Bretagne*, Great-Britain) still needs to be parsed and attached to the structure. The completed analysis is shown in (5c)

(5a) [TP [DP *L'* [NP *unification*]] [T' *bénéficiera*]]
[DP *la*]

(5b) [TP [DP *L'* [NP *unification*]] [T' *bénéficiera* [DP *la*]]]
Verb complementation error between *bénéficiera* (benefit-FUTURE) and *la* (the).

(5c) [TP [DP *L'* [NP *unification*]] [T' *bénéficiera* [DP *la* [NP *Grande-Bretagne*]]]]
Verb complementation error between *bénéficiera* (benefit-FUTURE) and *la* (the).[9]

3.2.3 Phonological reinterpretation

The underlying principles behind phonological reinterpretation at the spelling and grammatical levels are the same. However, at the grammatical level, the incorrect word is necessarily a word of the language, otherwise it would have been detected at the spelling level. Thus, phonological reinterpretation at the grammatical level is concerned with the diagnosis of the erroneous use of homophones, rather than of a wrong phoneme-to-grapheme rule.

[9] *La* (the) is indicated as the location of the error because it is the only element present in the erroneous phrase at the time the error is located, as shown in (5b).

These incorrect homophones can be detected only if they are sufficiently different from their correct counterpart. This "sufficiently different" is defined empirically as preventing a complete analysis of the input sentence because the troublesome word prevents the attachment of two partial structures together. In effect, in the absence of full semantic interpretation, words of different lexical categories are the best candidates, followed by incompatible agreement features (gender, number, or person), as well as distinct modes for verbs.

The input sentence is first diagnosed by constraint relaxation. If the result is not a full analysis of the sentence but a set of partial analyses, phonological reinterpretation is activated. Words at the border of the partial analyses are replaced by all their possible homophones. In order to figure out what they are, one retrieves from the lexicon the pronunciation of the suspicious word and one looks in the lexicon for all words with the same pronunciation. These phonetically similar words are added as alternatives into the parser and the sentence is parsed again. If a full analysis is reached with one of the alternative words, then a homophony error has been detected, and a correction proposal is available.

The authentic sentence extract (6a) is first analyzed as the three partial structures in (6b). Homophones for all the words located at the break between two partial analyses are looked up, that is for the words in (6c). However, homophones exist only for *ver* (worm), and they are very numerous as can be seen in (6d). *L'* as a clitic pronoun (it) has been taken into account earlier (during the lexical analysis preceding the first parse), because it has the same spelling as the elided determiner. The sentence is reanalyzed with all the alternative words in (6d). Only one complete analysis is reached, shown in (6e). Thus, *ver* (worm) is marked as a homophone error and *vers* (towards) is the proposed correction. Note that the lexical categories of the two homophones are different.

(6a) *Partir ver l'inconnu.*
Go worm the unknown

(6b) [TP[DP e][T' [VP *partir*]]]
[NP *ver*]
[DP *l'* [NP *inconnu*]]

(6c) *partir*, *ver*, *l'*

(6d) *vair*, *vairs*, *verre*, *verres*, *vers*, *vert*, *verts*.
vair, vairs, glass, glasses, worms/towards, green, green-plural

(6e) [TP[DP e][T' [VP *partir* [PP ***vers*** [DP *l'* [NP *inconnu*]]]]]]
go towards the unknown

3.2.4 Encountered difficulties

The most important difficulties encountered in the implementation of the grammar checker are directly linked to the techniques employed. Both constraint relaxation and phonological reinterpretation drastically increase the number of alternatives which must be treated by the system. This has several side effects. (i) Parsing takes more time to complete because each new alternative must be tried out. (ii) As a result of the increased number of alternatives, parsing often produces more complete analyses, out of which the most plausible one must be

sorted. (iii) As this sorting mechanism is not perfect, one sometimes displays the result of an analysis which is not the most adequate one and which can either fail to indicate some errors or contain occurrences of overflagging, that is, show errors on locations where there is none.

Example (7a) is perfectly grammatical. The singular agreement on the past participle *vu* (seen) forces the interpretation where it is the son which was seen. Constraint relaxation, however, allows the interpretation in (7b) where it is the neighbors which were seen, with the addition of a number error mark on the past participle. If the analysis in (7b) is selected for display, a perfectly grammatical sentence receives an error mark and thus contains an occurrence of overflagging.

(7a) *Le fils des voisins que j'ai vue est parti.*
The son of the neighbors whom I have seen is gone.

(7b) [TP[DP *le* [NP *fils* [PP *des* [DP [NP *voisins* [CP[DP *que*]$_i$[C [TP[DP *j'*][T *ai* [VP *vu* [DP e_i]]]]]]]$_i$]]]][T *est* [VP *parti*]]]
Number agreement error between *voisins* (neighbors), *que* (whom), and *vu* (seen).

A high number of overflagging occurrences is very detrimental to the system, as it then becomes impossible to trust the error messages of the diagnosis tool. Therefore, the number of overflagging occurrences must be kept to a minimum.[10] At the same time, though, the more true errors one detects, the higher the risk of overflagging. Thus, an appropriate balance between the number of true errors detected compared to the number of overflagging occurrences must be found.

3.3 Coherence checker

The coherence checker is designed to be used for the correction of exercises within a CALL software. Its aim is to verify that the answer given by the learner corresponds to the exercise question.

As the third step in the correction process, coherence checking assumes that the input sentence is free both of spelling errors and of grammar errors. From the syntactic analysis of the learner answer, an abstract representation of the sentence is created, called a pseudo-semantic structure (Etchegoyhen/Wehrle 1998). A pseudo-semantic structure contains semantic information on the sentence, without direct reference to its syntactic form. The exercise author has beforehand entered into the system one (and only one) possible answer for the question. A pseudo-semantic structure is also retrieved from the authorized answer. The two abstract representations are compared, and discrepancies are noted and reported.

Depending on the options selected by the exercise author, the system can tolerate a varying degree of difference between the two structures. For example, use of pronouns might be allowed or forbidden, active or passive voice may both be admitted. This gives some freedom to the language learners in the way they phrase their answers, while enabling the exercise author to enter only one possible correct answer, thus saving much time compared to the previous pattern matching answer coding techniques.

[10] This is achieved through a sentence ranking method privileging error free analyses.

The advantages of this coherence checker is that an incompatible answer will be rejected, while closely related ones will receive a detailed appropriate feedback without requiring from the exercise author the long and tedious task of entering many correct and incorrect solutions with their associated feedback.

Let us consider an illustration of the way the coherence checker functions for a model response entered by the author (8a) and the learner's responses in (8b) to (8d).

(8a) *Jean regarde le film à la télévision*.
Jean watches the movie on the television

(8b) *Il le regarde à la télévision*.
He it watches on the television.

(8c) *Jean regardait le film à la télévision*.
Jean watched the movie on the television

(8d) *Il pleut*.
It is-raining

If the use of pronouns has been enabled, (8b) will be accepted as correct by the system. Answer (8c) triggers a verb tense error feedback, as the imperfect tense is used instead of the present tense. Finally, a sentence like (8d) is rejected with a feedback saying that the answer is off-topic.

The coherence checker is thus a useful addition to the spell and grammar checkers to form a complete diagnosis tool, able to treat free-productions exercises and provide intelligent feedback for them, without being too time-consuming for the exercise authors.

4 Conjugation tool

The conjugation tool is typically a resource tool which users can consult whenever they feel like it. Its main purpose is to provide verb declensions for all the verbs of the French language. At first sight, it might seem that such an electronic tool does not offer more than its paper version. However, there are a number of advantages for providing such a tool online. First of all, instead of listing only some specific verbs with their full declension and having to refer to those when other verbs are asked for, an integrated version can provide the full declension of each verb with no additional cost.

On a paper version, verbs are accessible only through their infinitive form. This will not be helpful for a learner confronted with an unknown form of a very irregular verb. For instance, there is no possibility to deduce that *irai* (go-FUTURE) is the first singular person future form of the verb *aller* (go). However, by entering this inflected form into the conjugation tool, the learner is directed towards the declension of the full verb and is thus able to link the irregular form to its infinitive.

The electronic shape of the tool also allows the users to specify parts of the paradigm they are more particularly interested in, enabling the system to display only the relevant tenses and/or modes, thus preventing a possible overflow of information in the direction of the learners.

Moreover, some verb forms are ambiguous. A typical example is *suis* (am or follow) which can be either the first person singular present of the verb *être* (be), or the first/second person singular present of the verb *suivre* (follow). When presented with such an entry, the software presents both alternatives to the users and ask them to choose which infinitive they were looking for.[11]

The inner workings of the conjugation tool are rather simple, as full forms are stored in the system's lexicon. From a given input, the canonical form of the word, the infinitive in our case, is retrieved, possibly with the intervention of the user in case of an ambiguous verb form. The infinitive then gives access to all the verb forms associated with it in the lexicon. By sorting them into their appropriate person, tense and mode, one has already gathered the paradigm for simple tenses. Information associated with the canonical form contains the auxiliary (be or have) needed by the verb. This is used to recreate the composed tenses by juxtaposing the inflected auxiliary and the past participle. Once the whole verb paradigm is completed, it is made available through the interface which displays the requested information to the users.

The integration of the conjugation tool within a language learning software makes it easier to use than a paper version. There is no risk of forgetting it at home, it is always available and present, it does not get mislaid in a pile of books surrounding the computer. A few simple clicks brings it easily to the foreground, displaying right away the information looked for, without tedious searching for similar verbs in conjugation tables.

5 Conclusion

To be most effective, use of NLP tools within a CALL software must be designed with care. Giving access to NLP tools is not enough, especially as the target user population is not already familiar with them. Careful integration of the NLP tools into the didactic concept of the CALL software is a prerequisite to benefit plainly from this innovative technology.

All NLP tools, including the ones described above, have limitations. One might argue that this restricts their utility as language learning aids, because learners might be misled by some of the inadequate outputs provided by the NLP tools. A worst case scenario includes a risk for the users to mislearn rules of the target language. This, however, can only happen if the users rely completely on the NLP tools without being critical of them.

It is therefore of the utmost importance to warn the users of the limitations of the tools. Ideally, inadequate outputs provided by the NLP tools should make the learners reflect even more on the language they are learning and on its structure. Therefore, even NLP tool errors could help the learners master the target language because they require more thinking on their part.

Thus, NLP tools can be useful for CALL and usefully used in CALL. The present, imperfect state of the technology should not be a hindrance, although there is obviously room for improvement. However, improvements are often triggered by remarks and studies on how

[11] Hyperlinks to an online dictionary at this point would be a definite plus.

tools effectively work in context. Thus, if one waits to see improvements before using NLP tools in CALL software, one might have to wait for a long time, while using the tools in their present state will encourage improvements to be made according to the needs of the language learners.

Acknowledgments

Many thanks are due to Dr. Catherine Walther Green for her numerous and constructive comments on an earlier version of this paper. I also want to thank the two anonymous reviewers whose comments have helped improve this paper.

References

Abney, Steven (1987): *The English Noun Phrase in its Sentential Aspect*. PhD Dissertation. Cambridge, Mass.

Chomsky, Noam (1981): *Lectures on Government and Binding*. Dordrecht.

Courtin, Jacques/Dujardin, Danièle/Kowarski, Irène/Genthial, Damien/De Lima, Véra Lucia (1991): "Towards a complete detection/correction system". In: Nagao, Makato (ed.): *Proceedings of the International Conference on Current Issues in Computational linguistics*. Penang, Malaysia: 158-173.

Etchegoyhen, Thierry/ Wehrle, Thomas (1998): "Overview of GBGen: A large-scale, domain independent syntactic generator". In: Hovy, Eduard (ed.): *Proceedings of the 9th International Workshop on Natural Language Generation*. Niagara-on-the-Lake: 288-291.

Felshin, Sue (1995): "The Athena Language Learning Project NLP System: A multilingual System for Conversation-based Language Learning". In: Holland, V. Melissa/Kaplan, Johnathan D./Sams, Michelle R. (eds.): *Intelligent Language Tutors : Theory Shaping Technology*. Hove, UK: 257-272.

Gaudinat, Arnaud/Goldman, Jean-Philippe (1998): "Le système de synthèse FIPSVox: Syntaxe, phonétisation et prosodie". In: Bourlard, Hervé (ed.): *Actes des XXIIe journées d'études sur la parole*. Martigny, Suisse: 139-142.

Granger, Sylviane (2003): "Error-tagged Learner Corpora and CALL: A Promising Synergy". *CALICO Journal* 20/3: 465-480.

Heinecke, Johannes/Kunze, Jürgen/Menzel, Wolfgang/Schröder, Ingo (1998): "Eliminative Parsing with Graded Constraints". *Proceedings of Coling-ACL'98* 1: 526-530.

Holland, V. Melissa/Kaplan, Johnathan D./Sams, Michelle R. (eds.) (1995): *Intelligent Language Tutors : Theory Shaping Technology*. Hove, UK.

Laenzlinger, Christopher/Wehrli, Eric (1991): "FIPS: un analyseur interactif pour le français". *TA Information* 32/2: 35-49.

Matthews, Clive (1993): "Grammar Frameworks in Intelligent CALL". *CALICO Journal* 11: 5-27.

Menzel, Wolfgang/Schröder, Ingo (1998): "Error diagnosis for language learning systems". *Proceedings of NLP+IA'98* 1: 526-530.

Nerbonne, John (2003): "Natural language processing in computer-assisted language learning". In: Mitkov, R. (ed.): *The Oxford Handbook of Computational Linguistics*. Oxford: 670-698.

Peterson, James L. (1980): "Computer Programs for Detecting and Correcting Spelling Errors". *Communications of the ACM* 23/12: 676-687.

Pollock, Joseph J./Zamora, Antonio (1984): "Automatic spelling correction in scientific and scholarly text". *Communications of the ACM* 27/4: 358-368.

Schwind, Camilla (1995): "Error Analysis and Explanation in Knowledge Based Language Turoring". *Computer Assisted Language Learning Journal* 8/4: 295-324.

Schulze, Mathias/Hamel, Marie-Josée/Thompson, June (eds).(1999): *Language Processing in CALL*. ReCALL Special publication. Hull.

Swartz, M.L./Yazdani, M. (eds.)(1992): *Intelligent Tutoring Systems for Foreign Language Learning*. NATO ASI series F80. Berlin.

Vosse, Théo (1992): "Detecting and Correcting Morpho-Syntaxtic Errors in Real Texts". In: *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*. Trento, Italy: 111-118.

Wehrli, Eric (1997): *L'analyse syntaxique des langues naturelles : Problèmes et méthodes*. Paris.

Weischedel, Ralph M./Black, John E. (1980): "Responding Intelligently to Unparsable Inputs". *American Journal of Computational Linguistics* 6/2: 97-109.